# **QLIQ – AI Developer Job Test (Vector DB + LLMs)**

**Duration:** 5 days (120 hours)  
**Goal:** Build a functional AI Assistant that can onboard users, answer questions about the platform, and provide personalized suggestions using embeddings and LLM-powered responses.

## **What the Assistant Must Do**

1. **Conversational User Onboarding**
2. **Product/Service Recommendations** using embeddings
3. **Answer queries about the platform** (using RAG + vector DB)
4. **Basic personalized suggestions** based on user profile
5. **Simple API + Web Interface**

## **Features to Implement**

### **1. Conversational Onboarding Assistant (LLM + Prompting)**

* Guide users through interactive onboarding
* Ask key questions: user type (Influencer, Vendor, Buyer), interests, goals
* Generate a personalized onboarding summary using GPT or local LLM
* **Implement content filtering** - detect and handle inappropriate inputs
* **Bonus:** Add multilingual support (English + Arabic)

### **2. Vector Database for Contextual RAG**

**Create and index mock data into a Vector DB:**

* **Products catalog** (~50-100 mock products with descriptions)
* **Gig/Service descriptions** (~30-50 mock gigs)
* **Platform documentation** (FAQs, how-to guides, policies)
* **User guides** (how to use Qoyns, referral process, etc.)

**RAG Pipeline for queries like:**

* "How can I use my Qoyns?"
* "Show me trending gigs in electronics"
* "How do I refer someone?"

**Implementation:**

* Embed user query → Retrieve relevant chunks → Generate response using LLM
* **Stack Options:** Pinecone, Weaviate, Qdrant, Chroma, or local FAISS
* **Embeddings:** OpenAI (text-embedding-3-small), SentenceTransformers, or Cohere

### **3. Personalized Recommendations (Embedding Based)**

**Use similarity-based recommendation logic:**

* Embed user interests/profile → match with:
  + Products (for personalized suggestions)
  + Gigs (to suggest opportunities)
  + Other users (for network suggestions)
* Recommendations should adapt based on:
  + User's role (Vendor, Influencer, Buyer)
  + Stated interests and goals
  + Mock interaction history

**Implementation:** Cosine similarity or FAISS index for embeddings **Bias Detection:** Implement basic checks to ensure recommendations aren't skewed by protected characteristics

### **4. API + Simple Web Interface**

**REST API with endpoints:**

* POST /onboard – Accepts user data, returns onboarding summary
* POST /ask – Accepts query, responds via RAG
* GET /recommendations/{user\_id} – Returns personalized suggestions
* GET /health – Health check endpoint
* GET /metrics – Basic usage metrics (query count, response times)

**Web Interface:**

* Simple chat interface or form-based UI
* No need for complex styling – focus on functionality
* Should be publicly accessible

**Logging & Monitoring:**

* Log all API requests/responses with timestamps
* Track query types and response quality
* Basic error handling with informative messages

### **5. AI Model Training & Evaluation (NEW)**

**Custom ML Component:**

* Train a simple classification model to categorize user queries (FAQ, Product, Gig, General)
* Use scikit-learn or similar for basic ML implementation
* Evaluate model performance with accuracy, precision, recall metrics
* Compare performance against keyword-based classification

## **Mock Data Requirements**

**You must create realistic mock data for:**

### **Users (30-50 mock users):**

****{

"id": "user\_001",

"name": "Ahmed Hassan",

"type": "Influencer",

"interests": ["tech", "gadgets", "reviews"],

"location": "Dubai",

"qoyns\_balance": 150,

"network\_size": 1200

}

### **Products (50-100 mock products):**

****{

"id": "prod\_001",

"name": "Wireless Headphones",

"category": "Electronics",

"description": "Premium noise-cancelling wireless headphones...",

"price": 299,

"seller\_type": "Vendor"

}

### **Gigs (30-50 mock gigs):**

****{

"id": "gig\_001",

"title": "Product Review Video",

"description": "Create engaging product review videos...",

"category": "Content Creation",

"budget": 500,

"skills\_required": ["video editing", "social media"]

}

### **Platform Documentation:**

* Create 10-15 FAQ entries
* Write 5-10 how-to guides
* Include platform policies and rules
* Add Qoyns usage instructions

## **Deliverables**

1. **🌐 Live deployed link** (Vercel, Render, HuggingFace Spaces, etc.)
2. **📋 API Documentation** (Postman collection, Swagger, or simple README)
3. **💻 GitHub Repository** with:
   * Clear README with setup instructions
   * Mock data files or generation scripts
   * LLM and vector DB configuration
   * RAG pipeline implementation details
   * Model training code and evaluation results
   * Basic error handling and logging
   * AI ethics considerations document (bias mitigation, content filtering)

## **Evaluation Criteria**

| **Area** | **Weight** | **Details** |
| --- | --- | --- |
| **Vector DB + RAG** | 25% | Setup, retrieval accuracy, response relevance |
| **Recommendation Engine** | 20% | Personalized, embedding-based suggestions |
| **ML Model Training** | 20% | Query classification model, evaluation metrics |
| **Conversational UX** | 15% | LLM prompt design, onboarding flow |
| **API + Interface** | 10% | REST design, basic UI functionality |
| **Monitoring & Ethics** | 10% | Logging, bias detection, content filtering |

## **Suggested Stack (Flexible)**

| **Category** | **Options** |
| --- | --- |
| **LLMs** | OpenAI GPT-4/3.5-turbo, Claude, Mistral (via API)/ any of your choice |
| **Vector DB** | Pinecone, Qdrant, Chroma, or local FAISS |
| **ML Libraries** | scikit-learn, pandas, numpy (for classification model) |
| **Frameworks** | LangChain, LlamaIndex, or custom implementation |
| **Frontend** | React/Next.js, Vue, or simple HTML+JS |
| **Backend** | FastAPI, Flask, Node.js/Express |
| **Hosting** | Vercel, Render, Railway, HuggingFace Spaces |

## **Bonus Challenges (Optional)**

* **Multimodal AI:** Add image upload capability for product recommendations
* **Advanced ML:** Implement collaborative filtering for user-user recommendations
* **AI Safety:** Add advanced content moderation and toxicity detection
* **Voice Interface:** Add voice input/output capabilities
* **Real-time Features:** Implement WebSocket for live chat experience
* **A/B Testing:** Create simple A/B test framework for recommendation algorithms

## **Success Tips**

1. **Start with mock data generation** – create realistic, diverse datasets
2. **Focus on core functionality** over complex UI design
3. **Test your RAG pipeline** with various query types
4. **Train and evaluate your ML model** with proper metrics
5. **Implement basic bias detection** in recommendations
6. **Document your approach** and trade-offs made
7. **Include comprehensive logging** for debugging and monitoring
8. **Keep it simple but functional** – working features > complex non-working ones

## **What We're Looking For**

* **Functional AI implementation** with proper RAG pipeline
* **Custom ML model training** with evaluation metrics
* **Thoughtful mock data creation** that demonstrates understanding
* **Clear code organization** and comprehensive documentation
* **Working deployment** with accessible API and interface
* **AI ethics considerations** and bias mitigation strategies
* **Proper logging and monitoring** for production readiness
* **Evidence-based decision making** in AI model choices

**Remember:** This is about demonstrating your ability to build functional AI systems, not creating a perfect product. Focus on core functionality, clear implementation, and good documentation.